Package 'MOCK'

January 4, 2016

Type Package
Title Multiobjective Clustering With Automatic k-determination
Version 1.0
Date 2015-01-04
Author Dennis Assenmacher, Christian Homberg
Maintainer Dennis Assenmacher <dennis.assenmacher@uni-muenster.de>, Christian Homberg <c_homb01@uni-muenster.de></c_homb01@uni-muenster.de></dennis.assenmacher@uni-muenster.de>
Description MOCK implementation based on R. Provides the mock function as well as all related functions. Also contains various visualization functions.
License GPL (>= 2)
Depends ggthemes,ggplot2
Imports Rcpp (>= 0.11.6),ggplot2,ggthemes,RANN,emoa,gridExtra,nsga2R,vegan
LinkingTo Rcpp

R topics documented:

MOCK-package
binSearchC
connectivity
decodeC
deviationC
euclideanDistanceC
filterAndNormalizeFront
getNeighborhoodMutationMatrix
getNeighborhoodMutationMatrixC
meanVectorOfMatrixC
midpoint
mock
mockFunctionC
mutateC
mutateMock
NSGA2MOCK
printAttainmentScore
printClusterSolution
printMockRAndMocktoolFronts
printMultipleParetoFronts

2 binSearchC

What the pac	1 1	- (-1	1:		
					18
fDatasetCInteger					
fDatasetC					
				 	. 14
				 	Front

Description

More about what it does (maybe more than one line) $\sim\sim$ A concise (1-5 lines) description of the package $\sim\sim$

Details

Package: MOCK
Type: Package
Version: 1.0
Date: 2015-11-11
License: GPL (>= 2)

~~ An overview of how to use the package, including the most important functions ~~

Author(s)

Your Name

Maintainer: Your Name <your@email.com>

References

~~ Literature or other references for background information ~~

See Also

~~ Optional links to other man pages, e.g. ~~ ~~ <pkg> ~~

binSearchC

Find an integer in an ordered vector using binary search.

Description

Find an integer in an ordered vector using binary search.

connectivity 3

Usage

```
binSearchC(i, v)
```

Arguments

i Integer to be found in vector.

v IntegerVector ordered increasingly in which i is to be found.

Value

boolean indicating whether v contains i.

See Also

match

Examples

```
i = 42; v = 0:41
binSearchC(i,v)
```

connectivity

Calculate connectivity measure for a cluster and a corresponding neighborhood matrix.

Description

Calculate connectivity measure for a cluster and a corresponding neighborhood matrix.

Usage

```
connectivity(cluster, 1NN)
```

Arguments

cluster IntegerVector containing indexes of all points in currently considered cluster.

1NN IntegerMatrix containing the L nearest neighbors.

Value

connectivity measure according to definition in [Julia Handl 2004].

References

2004 Julia Handl - Multiobjective clustering with automatic determination of the number of clusters.

4 deviationC

decodeC

Decode the genetic representation of the edge assignment.

Description

Implementation is based on pseudo-code provided by [Julia Handl 2004].

Usage

```
decodeC(clusters)
```

Arguments

clusters

Integervector containing genetic representation of edges.

Value

Integervector containing the assigned cluster for each point.

References

2004 Julia Handl - Multiobjective clustering with automatic determination of the number of clusters.

Examples

```
clusters = c(2,3,1,5,4)
decodeC(clusters)
```

deviationC

Calculate the deviation within a matrix. This is defined as the sum of deviations of each point from the center vector of the matrix.

Description

Calculate the deviation within a matrix. This is defined as the sum of deviations of each point from the center vector of the matrix.

Usage

```
deviationC(m)
```

Arguments

m

NumericMatrix for which the deviation is to be calculated.

Value

deviation The sum of distances of each point from the center of ${\tt m}$

```
v = matrix(1:25,ncol=5)
deviationC(v)
```

euclideanDistanceC 5

euclideanDistanceC

Rcpp implementation of euclidean distance between two vectors.

Description

Rcpp implementation of euclidean distance between two vectors.

Usage

```
euclideanDistanceC(x, y)
```

Arguments

x First vector.y Second vector.

Value

Euclidean distance between input vectors x and y.

Examples

```
x = c(1, 3, 5); y = c(4, 3, 1)
euclideanDistanceC(x, y)
```

filter And Normalize Front

Filter and normalization of a paretro front

Description

Filter and normalization of a paretro front

Usage

```
\verb|filterAndNormalizeFront(front, minimumdev, maximumdev, minimumconn, maximumconn)|\\
```

Arguments

Minimumdev	minimum value for deviation
Maximumdev	maximum value for deviation
Minimumconn	Minimum value for connectivity
Maximumconn	Maximum value for connectivity

getNeighborhoodMutationMatrix

Calculates a neighborhood-matrix (square, but not symmetric) that contains likelihood for an edge (from row to column) to be mutated in PESA

Description

Calculates a neighborhood-matrix (square, but not symmetric) that contains likelihood for an edge (from row to column) to be mutated in PESA

Usage

getNeighborhoodMutationMatrix(nearestNeighbors)

Arguments

nearestNeighbors

Matrix that contains the nearest neighbors of each point ordered by neighborhood index

getNeighborhoodMutationMatrixC

Zero-based Rcpp implementation of which for integer vectors.

Description

Zero-based Rcpp implementation of which for integer vectors.

Usage

getNeighborhoodMutationMatrixC(nearestNeighbors)

Arguments

x Integer to be found.

v IntegerVector in which to find x.

Value

index Vector Integer Vector containing all zero-based indexes of appearences of x in v.

See Also

which

```
x = 42; v = 40:44
whichC(x,v)
```

mean Vector Of Matrix C 7

meanVectorOfMatrixC

Calculate the mean vector for a matrix by computing the mean for each column. This is an alternative to apply(m,2,mean).

Description

Calculate the mean vector for a matrix by computing the mean for each column. This is an alternative to apply(m,2,mean).

Usage

```
meanVectorOfMatrixC(m)
```

Arguments

m

NumericMatrix for which to calculate the mean vector.

Value

mean Numeric Vector containing the coordinates of the mean point of m.

See Also

mean

Examples

```
m = matrix(1:25,ncol=5)
meanVectorOfMatrixC(m)
```

midpoint

Calculate the (floor) mid of two integers.

Description

Calculate the (floor) mid of two integers.

Usage

```
midpoint(a, b)
```

Arguments

a First integer.b Second integer.

Value

Floor mid of a and b.

8 mock

mock Multiobjective clustering with automatic det of clusters.	termination of the number
--	---------------------------

Description

Implementation of MOCK as proposed by Handl.

Usage

```
mock(data, L = 10, ipMax = 10, epMax = 1000, gens = 1000,
  crossRate = 0.7, sigma = 0.5, nGrid = 10, maxCluster = NULL,
 method = "pesa", tournamentN = 2, printN = 10, distinctSolutions = F,
  controlFronts = 3, controlFrontMethod = "eigen", skipSearchBest = F)
```

Arguments

data Dataset on which to apply the MOCK algorithm.

L Amount of nearest neighbors to consider for initialization, mutation, and con-

nectivity objective function. Defaults to 10.

ipMax Maximum size of internal solutions used to explore new potential solutions.

Defaults to 10.

Maximum size of external population. Defaults to 1000. epMax

Number of generations. Defaults to 1000. gens Probability for parenting. Defaults to 0.7. crossRate

sigma Standard deviation for mutate function. Defaults to 0.5.

nGrid Number of grids. Defaults to 10.

maxCluster Basis for initialization. Corresponds to k_user of original MOCK parameters.

Default is dependent on chosen EMOA.

EMOA to apply. Defaults to "pesa". Other EMOAs are "nsga2" and "smsemoa". method

Size of tournament when applying tournament selection in NSGA-II. Defaults tournamentN

printN Intervals (in generations) in which to update the progress updates when running

the EMOAs. Simple one line progress bar, if NULL. Defaults to 10.

distinctSolutions

Boolean indicating whether NSGA-II and SMS-EMOA should omit duplicate solutions. Duplicates are removed within objective space. PESA-II doesn't add

duplicate solutions to pareto front. Defaults to false.

Amount of control fronts to create after applying the EMOA. Defaults to 3. controlFronts

controlFrontMethod

Method for creating the control data on which control fronts are based. Defaults to the method of creating uniform data in eigenspace of the original dataset proposed in 2007 by Handl. Other methods are "extrema" which creates points uniformly in original space of dataset and "" which creates points uniformly in unit hypercube.

skipSearchBest Boolean indicating whether search for best MOCK solution should be skipped. Defaults to false and is set to true when control fronts are generated.

mockFunctionC 9

Value

List of 9 items. \$sol allows to access the list of cluster solutions. Each cluster solution vector can be accessed with \$solution, the encoded clustering with \$param (can be decoded using decodeC), the amount of clusters with clusters, and the attainment score with \$score. The index of the best solution can be accessed via \$best. Also, some of the paramaters of the mock function call are included.

References

2004 Julia Handl - Multiobjective clustering with automatic determination of the number of clusters. 2007 Julia Handl - An Evolutionary Approach to Multiobjective Clustering

mockFunctionC

Objective MOCK function calculating the deviation and connectivity of a given genetic representation of a solution.

Description

Objective MOCK function calculating the deviation and connectivity of a given genetic representation of a solution.

Usage

```
mockFunctionC(data, geneOrClusterVector, 1NN, decoded)
```

Arguments

data NumericMatrix containing the whole dataset.

geneOrClusterVector

Integer Vector containing either the genetic representation of edges or the assign-

ment of points to clusters.

1NN IntegerMatrix containing the L nearest neighbors.

decoded boolean indicating whether geneOrClusterVector is already decoded. True

if geneOrClusterVector contains assignment of clusters to points, false if

geneOrClusterVector is genetic representation of edges.

Value

List consisting of the overall deviation and connectivity of the genes.

10 mutateMock

mutateC

Mutation of individuals in nearest neighbour space

Description

Mutation of individuals in nearest neighbour space

Usage

```
mutateC(sol, nn, neighborhood, randomVectorMutation,
  randomVectorNeighborSelection)
```

Arguments

sol: Solution to mutate

nn: Nearest neighbours matrix

neighborhood: Vector containing likelihood that an edge is mutated

randomVectorMutation:

Comparison vector for neighborhood

 ${\tt randomVectorNeighborSelection:}$

One of L nearest neighbors that is selected in case edge is mutated

 ${\it mutateMock}$

mutateMock Mutation operator for the MOCK algorithm

Description

 $mutate Mock\ Mutation\ operator\ for\ the\ MOCK\ algorithm$

Usage

```
mutateMock(param, nn, neighborhoodMutationMatrix, L)
```

Arguments

param Representation of one cluster solution

nn Nearest neighbour matrix

neighborhoodMutationMatrix

Matrix which contains the "costs" for each link i->j

Value

mutatedParam A mutated representatipn of one cluster solution

NSGA2MOCK 11

NSGA2MOCK	NSGA2 implementation	for MOCK algorithm.
NOONEHOON	1150112 implementation	joi moch argorum.

Description

NSGA2 implementation for MOCK algorithm.

Usage

```
NSGA2MOCK(gens = 100, fn, data, nn, neighborhoodMutationMatrix, population,
  L = 10, maxPop = 100, tournamentN = 2, printN = NULL,
  distinctSolutions = F)
```

Arguments

gens Number of generations.
fn Function to be applied.

data Data set on which to apply NSGA2MOCK.

nn Nearest neighbors matrix of data.

 $neighborhood {\tt Mutation Matrix}$

Neighborhood matrix which mock mutation is based on.

L Number of nearest neighbors to consider for mutation and mock function.

maxPop Maximum size of population.

tournamentN Size of tournament.

printN Generation interval in which to print progress of function. Progress bar is dis-

played, if null.

distinctSolutions

Boolean indicating whether to filter duplicate solutions in decision space.

initial Population

Initial population provided by mock wrapper.

Value

Population after gens generations. Matrix containing maxPop rows and nrow(data) + 2 columns. Last two columns contain connectivity and deviation.

References

 $2002\ Kalyanmoy\ Deb\ -\ A\ Fast\ Elitist\ Non-Dominated\ Sorting\ Genetic\ Algorithm\ for\ Multi-Objective\ Optimization:\ NSGA-II$

12 printClusterSolution

```
printAttainmentScore printClusterSolution
```

Description

Plots the attainment score for a distinct solution returned by the mock function.

Usage

```
printAttainmentScore(sol)
```

Arguments

sol

Solution returned by MOCK function.

Details

#'

Value

plot Attainment plot

Author(s)

Dennis Assenmacher

```
printClusterSolution printClusterSolution
```

Description

Draws a set of MOCK-Solutions in searchspace.

Usage

```
printClusterSolution(solution, index, data, columns)
```

Arguments

solution Solution returned by MOCK function.

index Index or vector of indices of MOCK solution/s.

data Original input data as a matrix or dataframe.

columns Determines the column count of the plot.

Details

#'

Author(s)

Dennis Assenmacher

```
printMockRAndMocktoolFronts
```

PrintCurrentSolutions

Description

Visualizes a 2-dimensional solution of PESA-II.

Usage

```
printMockRAndMocktoolFronts(mockSolution, mockToolSolution, absolute = "T",
   dataset = "")
```

Arguments

sol

Solution returned by MOCK function.

```
printMultipleParetoFronts
```

PrintMultipleParetoFronts

Description

Visualizes a 2-dimensional solution of PESA-II.

Usage

```
printMultipleParetoFronts(solutionList, absolute = "T", dataset = "")
```

Arguments

sol

Solution returned by MOCK function.

printParetoFront

PrintCurrentSolutions

Description

Visualizes a 2-dimensional solution of PESA-II.

Usage

```
printParetoFront(sol, method = "benchmark", maxClusters = 25,
  labelType = "index", gridIntervals = 0.1, dataset = "",
  markBestsolutions = T)
```

14 resize Vector

Arguments

sol Solution returned by MOCK function.

method Determines how the graph should be plotted. Options are: "benchmark", "fronts", "normalized", "unnormalized", "unnormal

maxClusters Determines which solutions should be visualized based on the cluster count.

labelType Determines whether the index of the cluster solution should be displayed within

the plot.

dataset Name of the dataset used in order to print the title.

markBestsolutions

Determines whether the best soultion should be displayed or not.

Value

ggplot to be created

resizeList

Resize an integervector to a given size.

Description

Resize an integervector to a given size.

Usage

```
resizeList(v, size)
```

Arguments

v IntegerVector to be resized. All elements with an index greather than size are

omitted.

size Integer determining the new seize of v.

Value

returnVector Resized IntegerVector v.

resizeVector

Resize an integervector to a given size.

Description

Resize an integervector to a given size.

Usage

```
resizeVector(v, size)
```

selectRowsOfDatasetC 15

Arguments

v IntegerVector to be resized. All elements with an index greather than size are

omitted.

size Integer determining the new seize of v.

Value

Resized IntegerVector v.

selectRowsOfDatasetC Select rows of a matrix based on a (one-based) index-vector.

Description

Select rows of a matrix based on a (one-based) index-vector.

Usage

```
selectRowsOfDatasetC(data, v)
```

Arguments

data A NumericMatrix on which a vector is to be applied.

v A one-based index-vector that is applied on data.

Value

dataSelection NumericMatrx containing only the selected rows of data.

See Also

Extract

```
data = matrix(1:25,ncol=5); v = c(1,4)
selectRowsOfDatasetC(data,v)
```

16 whichC

```
selectRowsOfDatasetCInteger
```

Select rows of a matrix based on a vector.

Description

Select rows of a matrix based on a vector.

Usage

```
selectRowsOfDatasetCInteger(data, v)
```

Arguments

data A NumericMatrix on which a vector is to be applied.

v A vector that is applied on data.

Value

dataSelection NumericMatrx containing only the selected rows of data.

See Also

base::Extract

Examples

```
data = matrix(1:25,ncol=5); v = c(1,4)
selectRowsOfDatasetC(data,v)
```

whichC

Zero-based Rcpp implementation of which for integer vectors.

Description

Zero-based Rcpp implementation of which for integer vectors.

Usage

```
whichC(x, v)
```

Arguments

x Integer to be found.

v IntegerVector in which to find x.

Value

index Vector Integer Vector containing all zero-based indexes of appearences of x in v.

whichCList 17

See Also

which

Examples

```
x = 42; v = 40:44
whichC(x,v)
```

whichCList

Zero-based Rcpp implementation of which for integer vectors.

Description

Zero-based Rcpp implementation of which for integer vectors.

Usage

```
whichCList(x, v)
```

Arguments

x Integer to be found.

v IntegerVector in which to find x.

Value

index Vector Integer Vector containing all zero-based indexes of appearences of x in v.

See Also

which

```
x = 42; v = 40:44
whichC(x,v)
```

Index

```
*Topic package
    MOCK-package, 2
<pkg>, 2
binSearchC, 2
connectivity, 3
decodeC, 4
deviationC, 4
euclideanDistanceC, 5
Extract, 15
filterAndNormalizeFront, 5
{\tt getNeighborhoodMutationMatrix}, {\tt 6}
{\tt getNeighborhoodMutationMatrixC}, {\tt 6}
match, 3
mean, 7
meanVectorOfMatrixC, 7
midpoint, 7
MOCK (MOCK-package), 2
mock, 8
MOCK-package, 2
mockFunctionC, 9
mutateC, 10
mutateMock, 10
NSGA2MOCK, 11
printAttainmentScore, 12
printClusterSolution, 12
\verb|printMockRAndMocktoolFronts|, 13
printMultipleParetoFronts, 13
printParetoFront, 13
resizeList, 14
resizeVector, 14
selectRowsOfDatasetC, 15
selectRowsOfDatasetCInteger, 16
which, 17
whichC, 16
whichCList, 17
```